This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

P.D. 12-10-1999
P. 1-20
20

XP-002194675

Maintaining integrity by assigning new busIDs on each bus reset as submitted to the p1394.1 committee

Dr. David V. James, Sony
3300 Zanker Road, MS SJ3C1
San Jose, CA 95134-4591
Phone: 408 955-6295
FAX: 408 955-4591
Email: dvj@alum.mit.edu

October 12 1999

This contribution is one of several, presented for independent review. An overall contribution, which provides the context for multiple contributions, is also provided in BR047R08.

Maintaining the integrity of existing devices mandates the disabling of remote-bus accesses after each bus reset. Otherwise, a connected node may observed unspecified state changes.

This proposal advocates the assignment of new bus_ID addresses after each bus reset. Not only does that ensure integrity, it eliminates the need to support virtual phyID assignments. Bridge processing of packets is then simplified, as illustrated in detailed tables.

Contribution to p1394.1 Working Group
Authored by Dr. David V. James, Sony.

NSDOCID: <XP___2194675A__I_>

High performance Serial Bus bridges

1. Introduction

1.13 Topology changes

1.13.1 Refresh and restart operations

Topology changes can change virtual-busId assignments, as new or reset buses receive previously unused addresses and old bus addresses are marked dirty. A cable-topology change starts with a bus reset on the affected bus, which forces that busId address to be reassigned. When the bus reset completes, portal-to-portal communications establish new busId-address assignments, in one of the following ways:

— Net refresh. A net refresh reassigns busId addresses to buses on the affected net. Previously assigned (but now unused) busId addresses are marked DIRTY.
— Net restart. A net restart assigns busId addresses to buses on the affected net. Existing EUI-to-nodeId address translations are purged. Previously assigned and DIRTY busId addresses are marked FREE.

The nature of the refresh/restart operation depends on the extent of the topology change and the survivor/victim classification of the nodes. The nonreset busId addresses of the survivors remain unchanged and the busId addresses of the victims (when necessary) are changed to avoid address-assignment conflicts. These address reassignment conventions are further detailed in the following subclauses.

The survivor portals are identified by the continued the presence identical prime-alpha and alpha portal identifiers; other portals are victim portals. Nodes managed by survivor portals are called survivor nodes; other nodes are called victim nodes. Survivor portal addresses are more stable, because they have precedence in the net-refresh and net-restart virtual address selection operations.

The prime-alpha and alpha portal identifiers are generated as a side effect of the net refresh and net restart operations (see xx). Each bus has one alpha portal and each net (collection of bridge-connected buses) has only prime-alpha portal. Survivor and victim classifications are based on EUI-64 identifiers, although supplemented EUI-64s are used to select between prime portal candidates.

1.13.2 Change notifications

A net reset is responsible for forcing the invalidation of EUI-to-virtualId translations that may be cached in victim nodes. Rather than relying on an unconfirmed broadcast invalidation signal, portals set quarantine bits to inhibit the use of possibly-stale EUI-to-virtualId translations.

A courtesy broadcast signal informs affiliated nodes of the change-of-topology condition. Since this is only a courtesy notification (no packets are corrupted if this is ignored), the system is not compromised by the less-robust nature of the unconfirmed broadcast indications. One form of courtesy notifications is provided, as follows:

— *bus_added*. One or more busIds have been added (when only one is added, that bus is identified).

JSDOCID: <XP__2194675A__I_>

1.13.3 Routing table properties

Each portal is assumed to have access to a 1024-entry routing table. Each routing table entry consists of two bits, which represent one the following states:

— FREE. Forwarding of this busId through portals is disabled.
The busId is not contained in any node's EUI-to-nodeId translation.
— DIRTY. Forwarding of this busId through portals is disabled.
The busId may remain in one or more EUI-to-nodeId translations.
— USED. The busId is not forwarded through this portal, but is forwarded through some others.
— FORW. The busId is forwarded through this portal (and possibly others in the busId-routing path).

The net refresh and restart operations assign virtualIds and leave bus-portal-resident routing tables initialized in a consistent fashion, as illustrated in figure 1. In this and following illustrations, the two-digit labels represent busId and localId portions of virtual addresses; the more relevant portal-resident routing tables are shown below their portal locations.

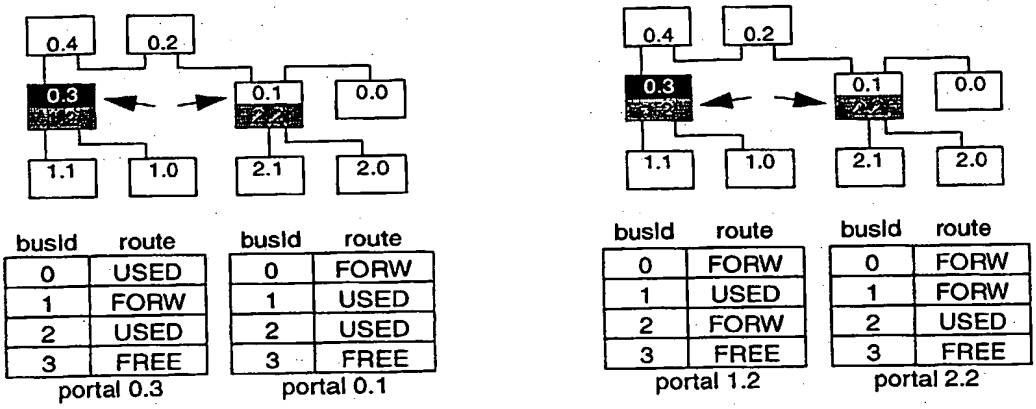


Figure 1—Routing table values (legend in 2.4.1)

The legend for figure 1 is used throughout this document and is described in 2.4. For the convenience of the reader, the legend notation is summarized here. The isolated rectangles and rectangle pairs represent nodes and bus-bridge portals respectively. In portals and nodes, the "2.1" label, '2' and '1' represent the *busId* and *localId* portions of the a virtual nodeId respectively. Black, grey, and white shadings represent prime-alpha, alpha, and non-alpha portals respectively.

NSDOCID: <XP___2194675A__I_>

The VALID name refers to a state that is either USED or FORW. When portal information is merged during the net reset or net refresh operations, these states are merged on a per-busId basis, using the merging specifiction algorithm defined in table 1.

Table 1—Validity merging algorithm

	Matching StateB			Differing StateB		
StateA	**VALID**	**DIRTY**	**FREE**	**VALID**	**DIRTY**	**FREE**
VALID	VALID	VALID	VALID	*DIRTIED*	*DIRTIED*	VALID
DIRTY	VALID	DIRTY	DIRTY	*DIRTIED*	*DIRTIED*	DIRTY
FREE	VALID	DIRTY	FREE	VALID	DIRTY	FREE

The left or right validity matrix is used if the contexts match or do not match respectively. For the entries marked *DIRTIED*, the resulting state is DIRTY and (due to potential use conflicts) this value cannot be used.

1.13.4 Bus topology changes

The bus is reset when a node is disconnected from the bus. The survivor subbus is identified by persistent prime-alpha and alpha portal identifiers; the other subbus is a victim. After the bus reset, the survivor and victim nodes perform bus refresh and bus restarts respectively. In all cases, the bus reset causes a new bus number to be assigned.

1.13.4.1 Disconnected node

In the survivor subbus portals, the reset bus is assigned a new *busId* (in this example *0* changes to *3*) is marked DIRTY and no quarantines are set. The victim node is responsible for detecting the absence of portals, which invokes a limited bus restart operation. The bus restart causes the affected node to discard its EUI-to-nodeId translations, as illustrated in figure 2. In this illustration, the tables represent validity maps for locally assigned stableId addresses.

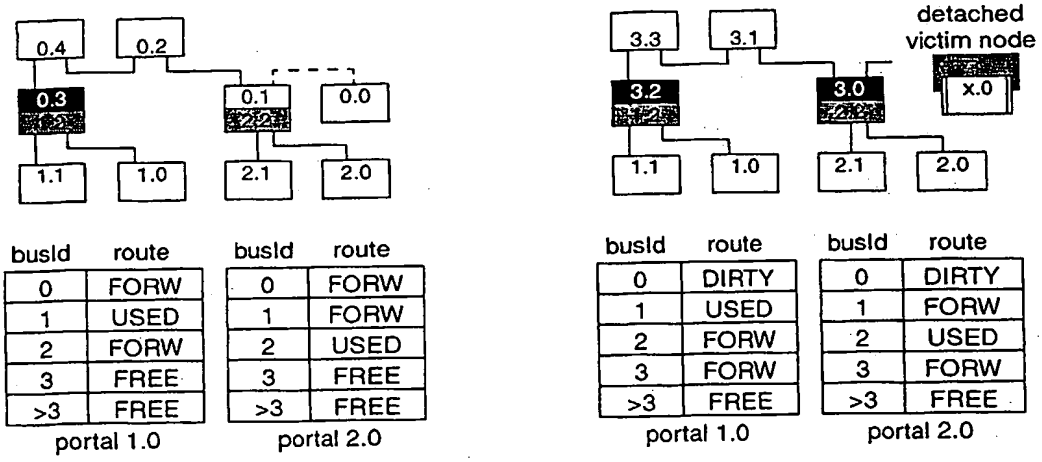


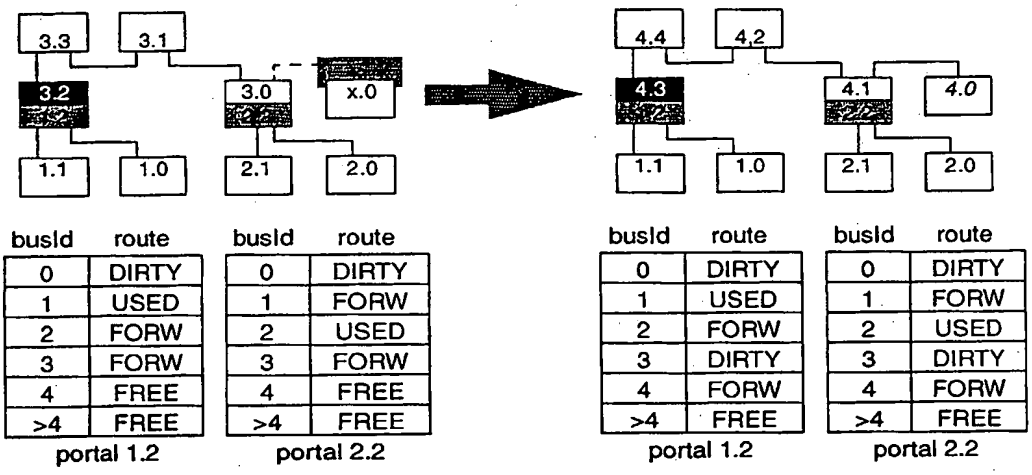
Figure 2—Disconnected node (legend in 2.4.1)

NSDOCID: <XP__2194675A__I_>

On the survivor buses, bus-1 and bus-2 nodes receive a courtesy *bus_added[3.2]* and *bus_added[3.0]* indications respectively, which allows other nodes to quickly determine how busId addresses have changed, as described in xx.

1.13.4.2 Reconnected node

When a node is reconnected to a bus, that bus is reset and assigned a new busId address, as illustrated in figure 3. As with the disconnection, bus-1 and bus-2 nodes receive courtesy *bus_added[4.3]* and *bus_added[4.1]* indications respectively.



busId	route
0	DIRTY
1	USED
2	FORW
3	FORW
4	FREE
>4	FREE

portal 1.2

busId	route
0	DIRTY
1	FORW
2	USED
3	FORW
4	FREE
>4	FREE

portal 2.2

busId	route
0	DIRTY
1	USED
2	FORW
3	DIRTY
4	FORW
>4	FREE

portal 1.2

busId	route
0	DIRTY
1	FORW
2	USED
3	DIRTY
4	FORW
>4	FREE

portal 2.2

Figure 3—Reconnected node (legend in 2.4.1)

1.13.4.3 Bus address assignments

A bus reset always causes a new busID address to be assigned, whether nodes are disconnected, connected, or unchanged. This inhibits others access of potentially inconsistent state (CSR locations and function may change during a bus reset). Assignment of new busID addresses also eliminates the possibility of unintentionally reassigning any node to the virtual addresses of another.

A bus reset forces local nodes' busID assignments to their initial $3FF_{16}$ value, as specified by the Serial Bus standard. The alpha portal is responsible for assigning nodes their new busID addresses after each bus reset, so that nodes can respond to virtual as well as physical addresses.

Address assignmens involve directed writes: the nonportals' NODE_IDS registers are written first and the portals' NODE_IDS registers are written last. Portals inhibit transmission of virtually addressed packets until their *NODE_IDS.busId* value changes from its initial $3FF_{16}$ value, so that packets are never sent to not-yet-initialized virtual adderesses.

NSDOCID: <XP___2194675A__I_>

1.13.4.4 Net topology changes

1.13.4.4.1 Subnet disconnection

When a subnet disappears from a net, net refresh and restart operations are performed on the survivor and victim subnets respectively. Survivor portals observe unchanged prime-portal and alpha-portal identifiers; other portals behave as victim portals. The victim-subnet portals are responsible for quarantining their bus-local nodes, forcing them to discard their EUI-to-virtualId translations, as illustrated in figure 4.

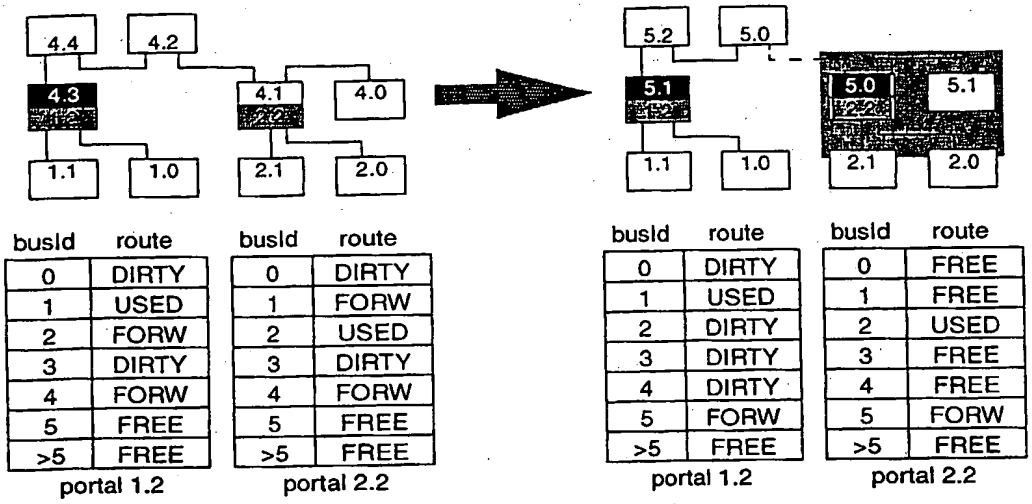


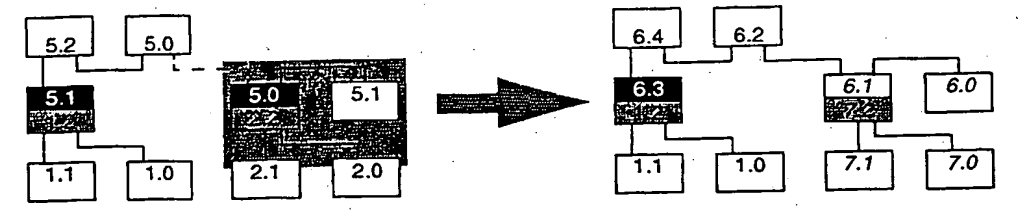
Figure 4—Subnet disconnection (legend in 2.4.1)

The bus-1 and bus-2 nodes receive courtesy *bus_added[5.1]* and *bus_added[5.0]* indications respectively.

NSDOCID: <XP___2194675A__I_>

1.13.4.5 Subnet reconnection

When a subnet is reconnected; the survivor subnet addresses remain unchanged. Previously disconnected subbus nodes are assigned new virtualIds and previously disconnected buses are assigned new busIds, as illustrated in figure 5.



busId	route
0	DIRTY
1	USED
2	DIRTY
3	DIRTY
4	DIRTY
5	FORW
6	FREE
7	FREE
>7	FREE

portal 1.2

busId	route
0	FREE
1	FREE
2	USED
3	FREE
4	FREE
5	FORW
6	FREE
7	FREE
>7	FREE

portal 2.2

busId	route
0	DIRTY
1	USED
2	DIRTY
3	DIRTY
4	DIRTY
5	DIRTY
6	FORW
7	FORW
>7	FREE

portal 1.2

busId	route
0	DIRTY
1	USED
2	DIRTY
3	DIRTY
4	DIRTY
5	DIRTY
6	FORW
7	USED
>7	FREE

portal 7.2

Figure 5—Subnet reconnection (legend in 2.4.1)

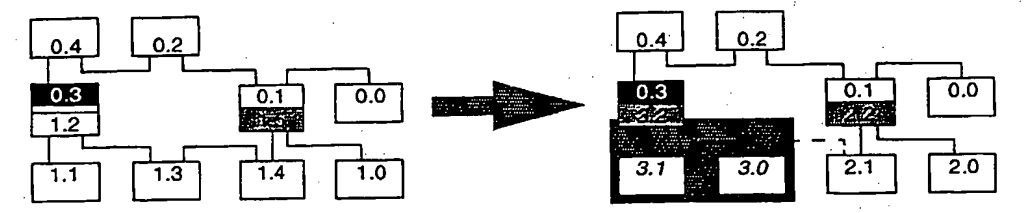
The bus-1 and bus-2 nodes receive courtesy *bus_added[6.3]* and *bus_added[6.1]* indications respectively.

3NSDOCID: <XP__2194675A__I_>

1.13.5 Redundant topology changes

1.13.5.1 Redundant disconnection

A redundant disconnection forces some virtual address changes, since previously merged subbuses receive distinct busId addresses, as illustrated in figure 6. A survivor subbus observes persistent prime-alpha and alpha portal identifiers; other subbuses are defined to be victims.



portal 1.2

busId	route
0	USED
1	USED
2	FREE
3	FREE
>3	FREE

portal 1.5

busId	route
0	FORW
1	USED
2	FREE
3	FREE
>3	FREE

portal 3.2

busId	route
0	FORW
1	DIRTY
2	FORW
3	USED
>3	FREE

portal 2.2

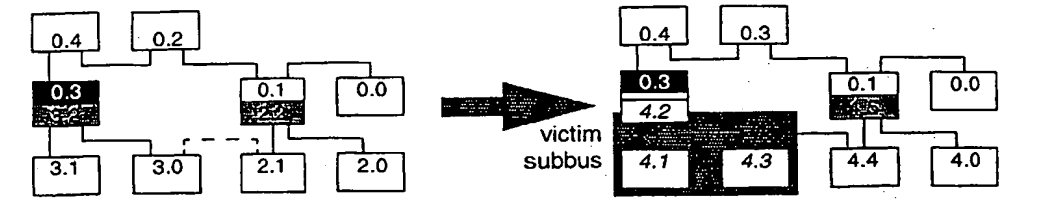
busId	route
0	FORW
1	DIRTY
2	USED
3	FORW
>3	FREE

Figure 6—Redundant disconnection (legend in 2.4.1)

Other buses receive *bus_added[3.2]* and *bus_added[2.2]* indications, allowing them to quickly respond to the revised busId-address assignments.

1.13.5.2 Redundant reconnection

A redundant reconnection forces some virtual nodeId changes, since distinct busId addresses are merged, as illustrated in figure 7. A survivor subbus is identified by a persistent prime-portal and alpha-portal identifiers; other subbuses are defined to be victims. The victim subbus inherits the busId of the survivor; victim nodes are assigned new (previously FREE) virtualId addresses.



portal 3.2

busId	route
0	FORW
1	DIRTY
2	FORW
3	USED
4	FREE
>4	FREE

portal 2.2

busId	route
0	FORW
1	DIRTY
2	USED
3	FORW
4	FREE
>4	FREE

portal 4.2

busId	route
0	USED
1	DIRTY
2	DIRTY
3	DIRTY
4	USED
>4	FREE

portal 4.5

busId	route
0	FORW
1	DIRTY
2	DIRTY
3	DIRTY
4	USED
>4	FREE

Figure 7—Redundant reconnection (legend in 2.4.1)

NSDOCID: <XP. _2194675A_I_>

Other buses receive a *bus_added[4.5]* indication allowing them to quickly respond to the revised busId-address assignments.

INSDOCID: <XP 2194675A I >

2. Packet routing

2.1 Asynchronous subaction routing

2.1.1 Subaction routing model

This subclause specifies the behavior of bridge portals when they receive and transmit requests (read, write, or lock) and responses. These subactions are accepted based on their *destination_ID* address, quarantined based on the *source_ID* address, and forwarded to the adjacent bus, as illustrated in figure 8..

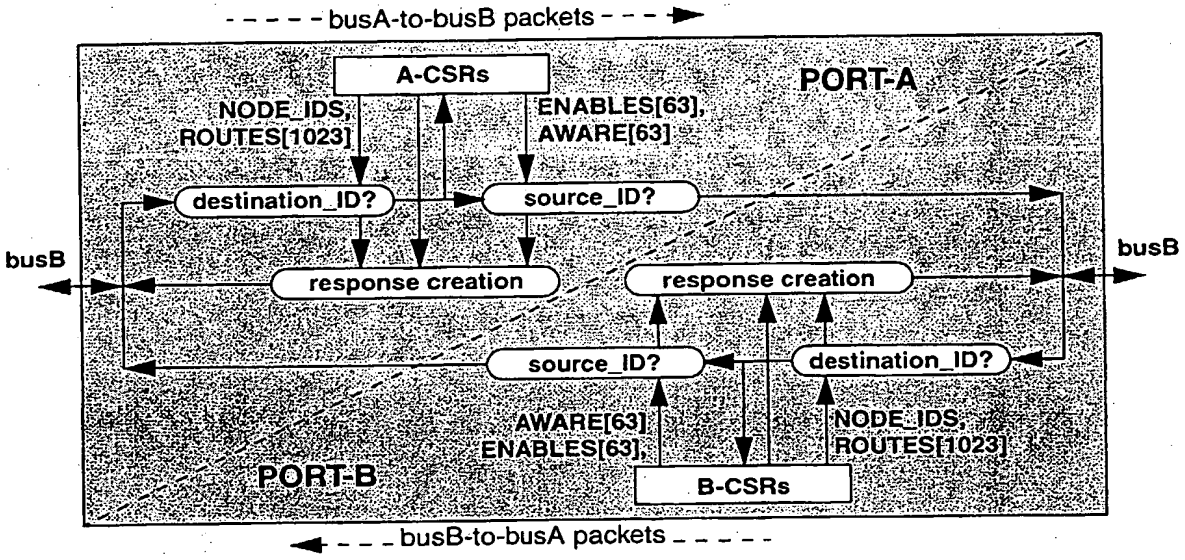


Figure 8—Asynchronous subaction processing

The request and response subactions that are forwarded in this fashion include the following: quadlet write request, block write request, write response, quadlet read request, block read response, lock request, and lock response.

There are 64 enable bits that (when at their initial zero value) quarantine local nodes by disabline the sending of request subactions from local-bus nodes, based on their *source_Id* identifier. Another set of 64 *bridgeAware* bits selectes between legacy and bridge-aware processing, also based on the packet's *source_ID* identifier.

When a source node is quarantined (its enable bit is zero), requests generated by that node are rejected with a quarantined indication. After a quarantined response indication is returned, a bridge aware node accepts the responsibility of discarding its (possibly stale) EUI-to-virtualId mappings before setting the bridge-resident enable bit and restarting the previously quarantined transaction.

A legacy device relies on the controller to clear its quarantined state, since legacy nodes were designed before this standard and are therefore not capable of restoring their quarantine state.

)NSDOCID: <XP___2194675A__I_>

2.1.2 Consumer portal processing

2.1.2.1 Packet consumption (destination_ID based)

A consuming portal eavesdrops on the bus and selectively accepts packets based on its *destination_ID* address. These packet are (in concept) labeled to simplify following less time critical processing steps, as specified in table 2. This table applies to packets with valid header_CRC values; other asynchronous packets shall be ignored..

Table 2—Packet consumption (destination_ID based)

Inputs					Row	Consumer processing	
type	delta	busID	localID	route[		label	acknowledge
GASP	—	—	—	—	1	THROUGH	—na—
not GASP	—	$3FF_{16}$	phyID	—	2	PORTAL	see table 3
		busId	phyID	—	3	VIRTUAL	
	1	busId	$3F_{16}$	—	4		
		$3FF_{16}$	badID	—	5	REJECT	
		busId	badID	—	6		
		diffId	—	FREE	7		
				DIRTY	8		
	—	diffId	—	FORW	9	THROUGH	

Notes:
route= ROUTES[destination_ID.bus_ID]
valid= SelfIDInfo[destination_ID.local_ID]
deltaPortal is 1 for delta portal, 0 otherwise

Row 1: GASP packets are accepted by all portals (excluding the transmitting portal).

Row 2: All portals shall accept physical-address packets directed to themselves.
Row 3: All portals shall accept virtual-address packets directed to themselves.
Row 4: The delta portal accepts packets directed to its aliased virtual-address.

Row 5: An delta portal shall accept packets to nonexistent fixed-bus addresses (these are later rejected).
Row 6: An delta portal shall accept packets to nonexistent local-bus addresses (these are later rejected).
Row 7: An delta portal shall accept packets to available remote-bus addresses (these are later rejected).
Row 8: An delta portal shall accept packets to retired remote-bus addresses (these are later rejected).

Row 9: A portals accepts packets that are destined for the adjacent bus.

JSDOCID: <XP 2194675A >

2.1.3 Consumer acknowledge codes

The acknowledge for accepted request and response subactions depends on the consumer's conditions when the packet is observed, as specified in table 3.

Table 3—Consumer acknowledge codes

type	Condition	Row	Acknowledge
—	A data_CRC or data_length error (primary failure)	1	ack_busy_X
		2	ack_data_error
	A data_CRC or data_length error (secondary failure)	3	ack_data_error
	Overly long (data_length>max_length)	4	ack_type_error
	Currently insufficient queue space	5	ack_busy_X, ack_busy_A, ack_busy_B
request	Available queue space	6	ack_pending
response	"	7	ack_complete

Row 1: Packets with corrupted payloads are initially discarded; an ack_busy_X should be returned.
Row 2: Packets with corrupted payloads are initially discarded; an ack_data_error may be returned.
Row 3: Packets with consistently corrupted payloads are discarded; an ack_data_error shall be returned.

Row 4: Packets with overly large data_length values are discarded; an ack_data_error shall be returned. In this context, overly large means larger than supported by bridge queue structures.

Row 5: Subactions are temporarily busied when the consuming portal has insufficient space. Busy indications specified by the 1394a dual-phase retry protocols shall be used.

Row 6: When request packets are accpeted and an ack_pending shall be returned.
Row 7: When request packets are accpeted and an ack_pending shall be returned.

MSDOCID: <XP___2194675A__I_>

2.1.4 Consumer portal relabeling (source_ID based)

After acceptance, the packet's source_ID determines how the accepted packets are processed at the consuming portal, as specified in table 4.

Table 4—Consumer portal relabeling

Inputs						Revised subaction label		
						Request		
label	firstCode	scope	aware	enabled	Row	Write	Nonwrite	Response
THROUGH	—	—	—	0	1	REJECTED		THROUGH
	firstRequest	—		1	2	PORTAL		—na—
	not firstRequest	local	0	1	3	POSTED	THROUGH	THROUGH
			1	1	4	THROUGH		THROUGH
		not local	—	—	5			
VIRTUAL	—	local	—	0	6	REJECTED		PORTAL
				1	7	PORTAL		
		not local	—	—	8			

Notes:
scope= (source_ID.bus_ID==NODE_IDS.bus_ID) ? local:!local;
enabled= ENABLE[source_ID.local_ID]
aware= AWARE[source_ID.local_ID]

Row 1: Virtual-address request packets are rejected when the local-bus source is quarantined.
Row 2: The FirstWrite and FirstRead requests are directed to the accepting portal.

Row 3: Nonquarantined legacy write requests are posted (create/return response and forward request).
Row 4: Nonquarantined bridge-aware requests and responses are delivered to the adjacent portal.
Row 5: Remotely sourced requests and responses are are delivered to the adjacent portal.

Row 6: Virtual requests for the portal are rejected when the local-bus source is quarantined.
Row 7: Nonquarantined requests and responses are delivered to the portal.
Row 8: Remotely sourced requests and responses are delivered to the portal.

3NSDOCID: <XP___2194675A__I_>

2.1.5 Producer portal processing

2.1.6 Producer portal routing

When the packet reaches the adjacent portal the packet's source_ID is checked to determine when the packet shall be discarded. Packets may be delivered to the portal CSRs, may be discarded, or may be delivered to the transmitter, as specified in table 5.

Table 5—Producing portal checking

Inputs								Row	Delivery location	
ready	tCode	ext	route	destBus	port	destPhy	aware		Request	Response
no	—	—	—	—	—	—	—	1	delayed	
yes	GASP	—	FORW	—	—	—	—	2	transmitter	
		—	not FORW	—	—	—	—	3	discarded	
	request	final	—	thisBus	—	—	—	4	portal CSRs	
	request, response	—	—	thisBus	delta	$3FF_{16}$	—	5		
					—	thisPhy	—	6		
						badPhy	0	7	rejected	discarded
						diffPhy	0	8	transmitter	discarded
				—				9	transmitter	

Notes:
#define ready (NODE_IDS.bus_ID==0X3FF)
#define route (ROUTES[source_ID.bus_ID])
#define destBus (destination_ID.bus_ID)
#define destPhy (destination_ID.local_ID)
#define thisBus (NODE_IDS.bus_ID)
#define thisPhy (NODE_IDS.local_ID)

Row 1: Subaction processing shall be delayed until *NODE_IDS.bus_ID* changes from its initial value.
Row 2: GASP packets are transmitted if their "response"would return in the opposing direction.
Row 3: GASP packets are discarded when passing through redundant routing paths.

Row 4: The FinalWrite and FinalRead requests are directed towards the destination-bus delta portal.
Row 5: The DeltaWrite/DeltaRead/DeltaSwap requests are directed to the destination-bus delta portal.

Row 6: Adjacent portal accesses shall be handled within the bridge and shall not appear on the bus.
Row 7: Nonexistent local-bus transactions are rejected internally and never appear on the bus.
Row 8: Returning response packets shall be discarded when addressed to legacy nodes.
Row 9: Other requests and responses are delivered to the adjacent bus.

NSDOCID: <XP___2194675A__I_>

Note that the GASP packet routing is based on the return route that would be taken by a "response", if one were to be generated. When compared to routing based on the initial spanning-tree on/off information, routing broadcast GASP packets in this fashion has several benefits:

1) Reduced costs. One set of routing tables is sufficient to route directed as well as broadcast packets.
2) Deadlock free. Any ROUTE[] tables that are deadlock free for directed asynchronous subactions are also known to be deadlock free for broadcast GASP deliveries.
3) Efficient. This forwarding algorithm remains efficient and consistent when the initial ROUTE[] tables are adjusted by reconfiguration software.

NOTE—Although no additional cycle start routing tables are required: the ROUTE[] tables must either be shared (which required multiplexing hardware) or replicated (which increases the amount of bridge storage).

2.1.7 Request acknowledge errors

When ready for transmission, the processing of the request depends on pre-transmission conditions and (if transmitted) the acknowledgment code that is returned, as specified in table 6 Whenever a response is generated by a bridge portal, the virtualId of the bridge shall be returned in the response; the location of this field is specified in 7.2.

Table 6—Subaction transmission processing

Condition	Row	request processing		response action
		rCode	sCode	
ack_pending	1	discard		discard, errorCount+= 1
ack_complete	2	resp_complete	observed ack	discard
ack_data_error	3	resp_data_error	"	truncated
ack_conflict_error	4	resp_conflict_error	"	discard errorCount+= 1
ack_tardy	5			
ack_type_error	6	resp_type_error		
ack_address_error	7	resp_address_error		
timer>timeLimit	8	resp_conflict_error	ext_delay_error	
missing acknowledge	9	discard, errorCount+=1		
ack_busy_X, ack_busy_A, ack_busy_B	10	retried		

Note:
timeLimit= (timeArrival +(request?Treq:Tres));

Row 1: An ack_pending indicates the request was accepted and can be discarded by the producer.
Row 2: An ack_complete completes the a write request or indicates the response was accepted.

Row 3: An ack_data_error terminates the transaction; a request changes to an error-reporting response. A response has its payload removed; resp_data_error and ext_data_error codes are inserted.

SDOCID: <XP__2194675A__I_>

Row 4: Row 5: Row 6: Row 7: An ack_conflict_error, ack_tardy, ack_type_error, or ack_address_error completes the transaction; the request is used to produce an error-reporting response.

Row 8: An excessive delay terminates the transmission; a request changes to error-reporting response. Note that a bridge portal uses Treq and Tres times to determine when the retries shall be terminated, not the Serial Bus defined retry-count and retry-time registers.

Row 9: A missing acknowledge causes the producer to discard request and response subactions.
Row 10: Request and response subactions are normally retried when busy indications are returned; dual-phase retry protocols shall be used.

3NSDOCID: <XP___2194675A__I_>

6.3.1 Pseudo acknowledge code valu s

When the response was generated by the bridge, rather than a nonbridge node, the *sCode* field provides additional transaction completion information, as specified in table 7.

Table 7—Bridge generated response-resident *scode* values

rCode	sCode	1394 ack-code name	Row	Description
resp_complete	1	ack_complete	2	Posted write completion
resp_conflict	B_{16}	ack_tardy	7	Terminated by acknowledge
	C_{16}	ack_conflict_error	8	
resp_data_error	D_{16}	ack_data_error	9	
resp_type_error	E_{16}	ack_type_error	10	
resp_address_error	F_{16}	ack_address_error	11	
resp_address_error	10_{16}	ext_node_error	12	Nonexistent node address
	11_{16}	ext_bus_error	13	Nonexistent bus address
resp_conflict	$1B_{16}$	ext_delay_error	15	Bridge delay termination
resp_data_error	$1C_{16}$	ext_quarantined	16	Quarantine termination
	$1D_{16}$	ext_posted_read	17	Legacy restricted
	$1E_{16}$	ext_truncated	18	Truncated response

Row 2: Remote transaction completion with an ack_complete.
Row 7: Remote transaction completion with an ack_tardy.
Row 8: Remote transaction completion with an ack_conflict_error.
Row 9: Remote transaction completion with an ack_data_error.
Row 10: Remote transaction completion with an ack_type_error.
Row 11: Remote transaction completion with an ack_address_error.

Row 12: The request's destinationId.localId corresponds to a existing virtual address.
Row 13: The request's destinationId.busId corresponds to a nonexistent virtual address.
Row 16: The source node's request was quarantined when it arrived at the first portal.
Row 17: The source node's read or lock request was rejected, due to legacy-device restrictions.
Row 15: A request exceeded its T_{req} residency-time restrictions after being accepted by a bridge.
Row 18: The response payload was stripped, due to an ack_data_error-indicated data_CRC error.

INSDOCID: <XP___2194675A__I_>

For all nonzero *sCode* values, the responder generating portal copies its virtualId into the 16-bit ***responder_ID*** field.

INSDOCID: <XP___2194675A__I_>

3NSDOCID: <XP___2194675A__I_>